Exercise 3.2: Grow the Cluster

> Open another terminal and connect into a your second node. Install **containerd** and Kubernetes software. These are the many, but not all, of the steps we did on the cp node.
>
> This book will use the **worker** prompt for the node being added to help keep track of the proper node for each command. Note that the prompt indicates both the user and system upon which run the command. It can be helpful to change the colors and fonts of your terminal session to keep track of the correct node.

1. Using the same process as before connect to a second node. If attending an instructor-led class session, use the same `.pem` key and a new IP provided by the instructor to access the new node. Giving a different title or color to the new terminal window is probably a good idea to keep track of the two systems. The prompts can look very similar.

2. `student@worker:~$ sudo -i`

3. `root@worker:~# apt-get update && apt-get upgrade -y`

   ```
   <If asked allow services to restart and keep the local version of software>
   ```

4. Install the containerd engine, starting with dependent software.

   ```
   root@worker:~#  apt install curl apt-transport-https vim git wget  \
   software-properties-common lsb-release ca-certificates  -y

   root@worker:~# swapoff -a

   root@worker:~# modprobe overlay

   root@worker:~# modprobe br_netfilter



   root@worker:~# cat << EOF | tee /etc/sysctl.d/kubernetes.conf
   net.bridge.bridge-nf-call-ip6tables = 1
   net.bridge.bridge-nf-call-iptables = 1
   net.ipv4.ip_forward = 1
   EOF

   root@worker:~# sysctl --system

   root@worker:~# mkdir -p /etc/apt/keyrings
   root@worker:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
   | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

   root@worker:~# echo \
     "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
     https://download.docker.com/linux/ubuntu \
     $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

   root@worker:~# apt-get update &&  apt-get install containerd.io -y
   root@worker:~# containerd config default | tee /etc/containerd/config.toml
   root@worker:~# sed -e 's/SystemdCgroup = false/SystemdCgroup = true/g' -i /etc/containerd/config.toml
   root@worker:~# systemctl restart containerd
   ```

5. Add Kubernetes repo

```
root@worker:~# vim /etc/apt/sources.list.d/kubernetes.list
```

```
    deb  http://apt.kubernetes.io/  kubernetes-xenial  main
```

6. Get the GPG key for the software

```
root@worker:~# curl -s \
        https://packages.cloud.google.com/apt/doc/apt-key.gpg \
        | apt-key add -
```

7. Update repos then install the Kubernetes software. Be sure to match the version on the cp.

```
root@worker:~# apt-get update
```

8.
```
root@worker:~# apt-get install -y \
        kubeadm=1.27.1-00 kubelet=1.27.1-00 kubectl=1.27.1-00
```

9. Ensure the version remains if the system is updated.

```
root@worker:~# apt-mark hold kubeadm kubelet kubectl
```

10. Find the IP address of your **cp** server. The interface name will be different depending on where the node is running. Currently inside of **GCE** the primary interface for this node type is ens4. Your interfaces names may be different. From the output we know our cp node IP is 10.128.0.3.

```
student@cp:~$ hostname -i
```

```
    10.128.0.3
```

```
student@cp:~$ ip addr show ens4 | grep inet
```

```
    inet 10.128.0.3/32 brd 10.128.0.3 scope global ens4
    inet6 fe80::4001:aff:fe8e:2/64 scope link
```

11. At this point we could copy and paste the **join** command from the cp node. That command only works for 2 hours, so we will build our own **join** should we want to add nodes in the future. Find the token on the cp node. The token lasts 2 hours by default. If it has been longer, and no token is present you can generate a new one with the **sudo kubeadm token create** command, seen in the following command.

```
student@cp:~$ sudo kubeadm token list
```

```
    TOKEN                      TTL       EXPIRES                  USAGES
        DESCRIPTION                                   EXTRA GROUPS
    bml44w.3owxl50rrtymamt7   2h        2023-05-27T18:49:41Z    authentication,signing
        <none>                                        system:bootstrappers:kubeadm:default-node-token
```

12. We'll assume you are adding a node more than two hours later and create a new token, to use as part of the join command.

```
student@cp:~$ sudo kubeadm token create
```

```
    27eee4.6e66ff60318da929
```

13. Create and use a Discovery Token CA Cert Hash created from the cp to ensure the node joins the cluster in a secure manner. Run this on the cp node or wherever you have a copy of the CA file. You will get a long string as output. Also note that a copy and paste from a PDF sometimes has issues with the caret (^) and the single quote (') found at the end of the command.

   THE LINUX FOUNDATION | Training & Certification

```
student@cp:~$ openssl x509 -pubkey \
        -in /etc/kubernetes/pki/ca.crt | openssl rsa \
        -pubin -outform der 2>/dev/null | openssl dgst \
        -sha256 -hex | sed 's/^.* //'
```

```
6d541678b05652e1fa5d43908e75e67376e994c3483d6683f2a18673e5d2a1b0
```

14. On the **worker node** add a local DNS alias for the cp server. Edit the `/etc/hosts` file and add the cp IP address and assign the name `k8scp`. The entry should be exactly the same as the edit on the cp.

```
root@worker:~# vim /etc/hosts


10.128.0.3 k8scp      #<-- Add this line
127.0.0.1 localhost
....
```

15. Use the token and hash, in this case as `sha256:long-hash` to join the cluster from the **second/worker** node. Use the **private** IP address of the cp server and port 6443. The output of the **kubeadm init** on the cp also has an example to use, should it still be available.

```
root@worker:~# kubeadm join \
    --token 27eee4.6e66ff60318da929 \
    k8scp:6443 \
    --discovery-token-ca-cert-hash \
    sha256:6d541678b05652e1fa5d43908e75e67376e994c3483d6683f2a18673e5d2a1b0
```

```
    [preflight] Running pre-flight checks


    [preflight] Reading configuration from the cluster...
    [preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm
    ↪   kubeadm-config -oyaml'
    [kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
    [kubelet-start] Writing kubelet environment file with flags to file
    ↪   "/var/lib/kubelet/kubeadm-flags.env"
    [kubelet-start] Activating the kubelet service
    [kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...


    This node has joined the cluster:
    * Certificate signing request was sent to apiserver and a response was received.
    * The Kubelet was informed of the new secure connection details.


    Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

16. Try to run the **kubectl** command on the secondary system. It should fail. You do not have the cluster or authentication keys in your local `.kube/config` file.

```
root@worker:~# exit
```

```
student@worker:~$ kubectl get nodes
```

```
    The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

```
student@worker:~$ ls -l .kube
```

```
    ls: cannot access '.kube': No such file or directory
```